

Prozedurale Generierung von Baumstrukturen innerhalb der Unreal Engine 4

Procedural generation of tree-like structures in the Unreal Engine 4

David Liebemann

Bachelor-Abschlussarbeit

Betreuer:

Prof. Dr. Christof Rezk-Salama

Trier, 26.02.2017

Kurzfassung

Inhaltsverzeichnis

1	Einleitung und Problemstellung	1
1.1	Prozedurale Generierung	1
1.2	Bisherige Arbeiten	1
1.3	Ansatz	1
1.4	Unreal Engine 4	1
2	Lindenmayer-Systeme	2
2.1	Kontextfreie Grammatik	2
2.2	D0L-Systeme	3
2.2.1	Parametrische L-Systeme	4
2.3	Grafische Interpretation von L-Systemen	5
2.3.1	Turtle-Interpretation	6
2.3.2	Verzweigte L-Systeme	7
2.3.3	Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum	8
2.4	Anpassungen für Baumstrukturen	11
2.4.1	Graphentheoretische Bäume	11
2.4.2	Tropismus	13
3	Space Colonization Algorithmus	15
3.1	Ursprung	15
3.2	Aufbau	15
3.3	Ablauf	16
3.4	Aufbau der Baumstrukturen	18
3.5	Erweiterungen	18
4	Implementierung	22
4.1	Baumrepräsentation	22
4.2	L-Systeme	23
4.2.1	Parameter	23
4.2.2	Ableitung	24
4.2.3	Turtle Interpretation	24
4.3	Space-Colonization Algorithmus	24

4.3.1	Parameter	25
4.3.2	Einflussbereich	25
4.3.3	Ablauf des Algorithmus	25
4.4	Modellgenerierung	25
4.4.1	Parameter	25
4.4.2	Procedural Mesh Component	25
4.4.3	Generierung der Zylinder-Meshes	25
4.4.4	Operationen auf der Baumstruktur	26
5	Ergebnisse	27
5.1	L-System-Akteur	27
5.1.1	Monopodial	27
5.1.2	Simpodial	27
5.1.3	Ternäre Verzweigungen	27
5.1.4	Tropismus	27
5.1.5	Performanz	27
5.2	Space-Colonization-Akteur	27
5.2.1	Ursprüngliche Parameter	27
5.2.2	Erweiterte Parameter	28
5.2.3	Performanz	28
5.3	Probleme	28
6	Zusammenfassung und Ausblick	29
6.1	Erweiterungen	29
6.1.1	Space Colonization Algorithmus	29
6.1.2	Aststruktur	29
6.1.3	Texturen	29
6.1.4	Blätter	29
6.1.5	Generierung zur Laufzeit	29
6.1.6	Verteilung	30
6.1.7	Instanziierung	30
6.1.8	title	30
	Literaturverzeichnis	31
	Glossar	32
	Erklärung der Kandidatin / des Kandidaten	33

Einleitung und Problemstellung

test

1.1 Prozedurale Generierung

Insbesondere Vegetation - warum, was sind die Schwierigkeiten, wenn man es nicht prozedural generiert.

1.2 Bisherige Arbeiten

test

1.3 Ansatz

Verwendete Ansätze: L-Systeme mit Turtle-Graphik und Space Colonization Algorithmus. Kleine Einleitung zu beiden Ansätzen, bisherige Verwendung.

Beschreibung - was wird mit Baumstruktur / Baum-Graph gemeint?

1.4 Unreal Engine 4

Allgemeine Erklärung - was ist die Unreal Engine 4, was stellt sie zur Verfügung, wie entwickelt man dafür? Insbesondere Eingabe von Parametern über den Editor -> wichtig für Implementierung.

Erklärung - was ist ein Unreal-Akteur? Was sind die Komponenten, die diesem hinzugefügt werden können? Sämtliche Visualisierungen von L-Systemen wurden mithilfe des Unreal Engine 4 Projekts erstellt.

Erklärung - was ist ein L-System-Akteur, was ist ein Space Colonization-Akteur

Alle Längenangaben entsprechen der in der Unreal Engine 4 verwendeten Längeneinheit *cm*.

Lindenmayer-Systeme

Es werden die Regeln für die Definition von Lindenmayer-Systemen – kurz: L-Systeme – festgelegt und die verwendete Methode zur Visualisierung der Ergebnisse von L-Systemen besprochen.

2.1 Kontextfreie Grammatik

Bei L-Systemen handelt es sich um, auf Zeichenketten basierende, Ersetzungssysteme. Es werden komplexe Objekte beschrieben, indem Teile der Zeichenkette durch andere Zeichen oder Zeichenketten ersetzt werden. Die Beschreibung dieser Ersetzungen findet mittels festgelegter Produktionsregeln statt. [PL04, S.2]

Eine formale Definition eines auf Zeichenketten arbeitenden Ersetzungssystems wird durch kontextfreie Grammatiken gegeben:

Kontextfreie Grammatik: 2.1 *Eine kontextfreie Grammatik G ist ein Tupel $G = (V, N, P, \omega)$ bestehend aus:*

V *Einer nichtleeren, endlichen Menge von Buchstaben (Alphabet).*

N *Einer endlichen Menge von Variablen.*

P *Einer endlichen Menge von Produktionsregeln in der Form $P : A \rightarrow \alpha$ mit $A \in N$ und $\alpha \in (V \cup N)^*$.*

$\omega \in N$ *Dem Axiom, Startsymbol der Grammatik.*

[Sch14, S.343]

Die Menge V^* ist die Menge aller Wörter über V – die Menge aller Wörter, die aus dem Alphabet V gebildet werden können. [Sch14, S.70]

Eine Grammatik wird als kontextfrei bezeichnet, wenn beispielsweise die Produktionsregel $A \rightarrow \alpha$ angewendet werden kann, ohne die A umgebenden Buchstaben – seinen Kontext – beachten zu müssen. [Sch14, S.343]

Eine Grammatik wird als deterministisch bezeichnet, wenn es genau eine Produktionsregel $r \in P$ für jede Variable $A \in V$ gibt, sodass $r : A \rightarrow \alpha$, $\alpha \in (V \cup N)^*$. Das bedeutet, dass die Ersetzung einer Variable eindeutig durch eine einzige Regel beschrieben wird. [STN16, S.75]

Die Anwendung der Produktionsregeln findet meist sequentiell statt – die Zeichenkette wird von links nach rechts durchlaufen und Ersetzungen werden direkt auf die untersuchte Zeichenkette angewendet. [STN16, S.75]

2.2 D0L-Systeme

Diese Arbeit beschränkt sich auf die Behandlung deterministischer, kontextfreier L-Systeme, auch D0L-Systeme genannt. Diese besitzen die Eigenschaften einer deterministischen und kontextfreien Grammatik, Produktionsregeln werden jedoch parallel und gleichzeitig auf alle Buchstaben des untersuchten Wortes angewendet. Dieses Vorgehen soll die Zellteilung in mehrzelligen Organismen simulieren und ist somit an biologische Vorgänge angelehnt. [PL04, S. 3]

Ein D0L-System kann wie folgt definiert werden:

D0L-System: 2.1 *Ein D0L-System ist ein Tupel $G = (V, P, \omega)$, bestehend aus:*

V *Einem nichtleeren, endlichen Alphabet.*

P *Einer endlichen Menge von Produktionsregeln in der Form $P : a \rightarrow b$ mit $a \in V$ und $b \in V^*$. a wird als Vorgänger, b als Nachfolger bezeichnet. Ist für einen Buchstaben $x \in V$ keine explizite Produktionsregel angegeben, wird die Identitätsproduktion $P : x \rightarrow x$ angenommen – der Buchstabe wird durch sich selbst ersetzt.*

$\omega \in V^+$ *Dem Axiom, Startsymbol der Grammatik.*

[PL04, S.4]

Die Menge V^+ ist die Menge aller nichtleeren Wörter über V .

Die Ableitung eines Wortes entspricht der Ersetzung aller Buchstaben anhand der Produktionsregeln. Ein Wort kann mehrmals abgeleitet werden.

Ableitung: 2.1 *Gegeben sei ein Wort $w = a_1 \dots a_m$ mit $w \in V^*$ und $a_i \in V^*$. Das Wort $v = b_1 \dots b_m$ mit $v \in V^*$ und $b_i \in V^*$ ist die Ableitung von w wenn für alle $i = 1 \dots m$ eine Produktionsregel $a_i \rightarrow b_i$ existiert. Die Ableitung wird als $w \Rightarrow v$ notiert.*

Das Wort w_n ist die n -te Ableitung des Wortes w_0 wenn eine Folge von Wörtern w_0, w_1, \dots, w_n mit Ableitungen $w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n$ existiert. [PL04, S.4]

Beispiel: Das Wachstum der Blaualgen-Gattung „Anabaena“ kann durch ein L-System simuliert werden. Die Buchstaben a und b beschreiben die Größe und Teilungsbereitschaft einer Algenzelle, während die Indizes l und r die Polarität einer Zelle darstellen. Es gelten folgende Produktionsregeln:

$$\begin{aligned}
p_1 : a_r &\rightarrow a_l b_r \\
p_2 : a_l &\rightarrow b_l a_r \\
p_3 : b_r &\rightarrow a_r \\
p_4 : b_l &\rightarrow a_l
\end{aligned} \tag{2.1}$$

Die Entwicklung einer Anfangszelle a_r (Axiom $\omega : a_r$) läuft daraufhin wie in Abbildung 2.1 dargestellt ab.

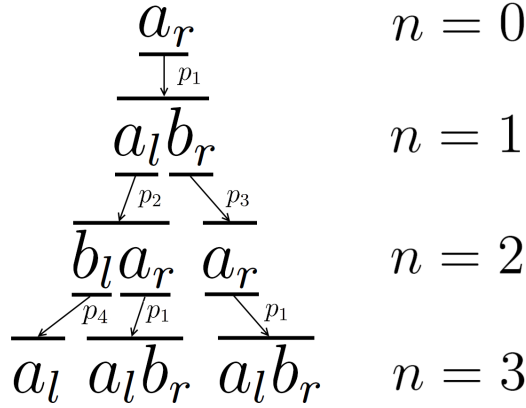


Abb. 2.1: Die n-fache Ableitung der Anfangszelle a_r anhand der Produktionsregeln $p_1 \dots p_4$ aus Gleichung 2.1. Eigene Abbildung auf Basis von [PL04, S.4].

2.2.1 Parametrische L-Systeme

Parametrische L-Systeme stellen eine Erweiterung der D0L-Systeme dar. Die Buchstaben eines verwendeten Alphabets V werden um zugeordnete Parameter aus der Menge der reellen Zahlen ergänzt. Ein solches parametrisches Wort $V \times \mathbb{R}^*$ besteht aus einem Zeichen $A \in V$ und Parametern $a_1, \dots, a_n \in \mathbb{R}$ und wird als $A(a_1, \dots, a_n)$ dargestellt. Ein parametrisches Wort ohne Parameter mit dem Zeichen $A \in V$ wird schlicht als A dargestellt. [PL04, S.41]

Während in der Definition von Modulen numerische Konstanten verwendet werden, werden bei der Angabe eines L-Systems Variablen verwendet. Ist Σ eine Menge von Variablen, dann ist $E(\Sigma)$ ein arithmetischer Ausdruck, in dem Variablen, Konstanten und arithmetische Operatoren auf eine zulässige Weise kombiniert werden. [PL04, S.41]

Parametrisches L-System: 2.2.1 *Ein Parametrisches L-System ist ein Tupel $G = (V, \Sigma, P, \omega)$, bestehend aus:*

V *Einem nichtleeren, endlichen Alphabet.*

Σ *Einer Menge von Variablen.*

P Einer endlichen Menge von Produktionsregeln $P : (V \times \Sigma^*) \rightarrow (V \times E(\Sigma))^*$

$\omega \in M^+$ mit $M = (V \times \mathbb{R}^*)$ – einem Axiom in Form eines nichtleeren, parametrischen Wortes.

[PL04, S.41]

Eine Produktion kann auf ein parametrisches Wort angewendet werden wenn das Zeichen, das dem Wort vorausgeht, und die Anzahl der Parameter mit dem Zeichen und der Anzahl der Parameter im Vorgänger der Produktionsregel übereinstimmen. [PL04, S.42]

Beispiel: Gegeben sei folgendes, parametrisches L-System:

$$\begin{aligned} \omega &: A(1, 1) \\ p_1 &: A(x, y) \rightarrow A(x + 1, y * 2) \ B(y) \\ p_2 &: B(x) \rightarrow B(x + 1) \ C \end{aligned} \quad (2.2)$$

Das Alphabet V und die Menge der Variablen Σ gehen implizit aus der Angabe der Produktionsregeln hervor und werden in zukünftigen L-System-Gleichungen nicht angegeben. Die Entwicklung des L-Systems läuft anhand Abbildung 2.2 ab.

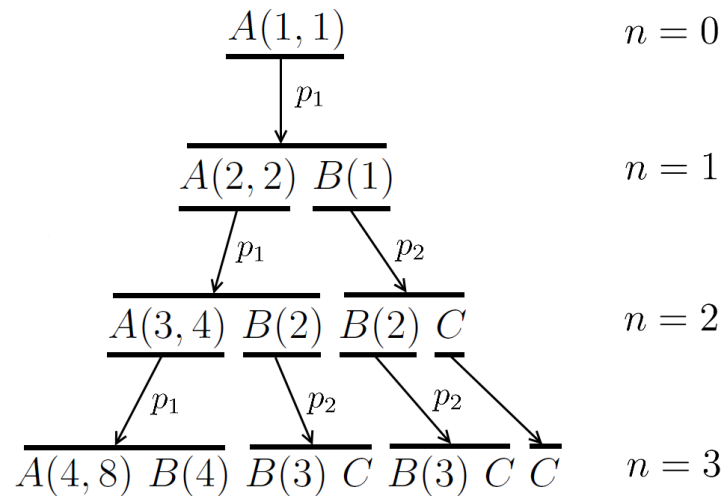


Abb. 2.2: Die n-fache Ableitung des Axioms $A(1, 1)$ anhand der Produktionsregeln aus Gleichung 2.2. Die Ersetzung des Zeichens C erfolgt anhand der impliziten Identitätsproduktion. Eigene Abbildung.

2.3 Grafische Interpretation von L-Systemen

Um die Ergebnisse von L-Systemen in Form von dreidimensionalen Objekten zu visualisieren, muss eine grafische Interpretation der resultierenden Zeichenket-

ten festgelegt werden. Im Folgenden wird die verwendete Visualisierungsmethode, Turtle-Interpretation, vorgestellt.

2.3.1 Turtle-Interpretation

Die Turtle-Interpretation im zweidimensionalen Raum entspricht der Vorstellung einer Turtle¹ auf einem Blatt Papier. Die Turtle besitzt eine Position \vec{p} sowie einen Einheitsvektor \vec{H} in kartesischen Koordinaten. \vec{p} beschreibt die Position der Turtle auf der Ebene und \vec{H} entspricht der Blickrichtung (Heading) der Turtle. Der Zustand einer Turtle wird somit vollständig durch die Position und Blickrichtung definiert und wird als Tupel (\vec{p}, \vec{H}) angegeben. [GSJ04, S.2] Die Ausgangsposition entspricht dem Ursprung des lokalen Koordinatensystems der Turtle.

Es können drei Aktionen durchgeführt werden, welche durch die folgenden Symbole dargestellt werden:

$F(l)$ Die Turtle bewegt sich um $l > 0$ in die Richtung der aktuellen Blickrichtung. Die neue Position ist \vec{p}_{neu} mit:

$$\vec{p}_{neu} = \vec{p} + \vec{H} * l \quad (2.3)$$

Zwischen der alten Position \vec{p} und der neuen Position \vec{p}_{neu} wird eine Linie gezeichnet.

$+(d)$ Die Turtle dreht sich um den Winkel d nach links. Die neue Blickrichtung ist \vec{H}' mit:

$$\vec{H}' = \begin{pmatrix} \cos(d) & -\sin(d) \\ \sin(d) & \cos(d) \end{pmatrix} * \vec{H} \quad (2.4)$$

$-(d)$ Die Turtle dreht sich um den Winkel d nach rechts. Die neue Blickrichtung ist \vec{H}' mit:

$$\vec{H}' = \begin{pmatrix} \cos(d) & \sin(d) \\ -\sin(d) & \cos(d) \end{pmatrix} * \vec{H} \quad (2.5)$$

[GSJ04, S.4,46] [PL04, S.7] Die Symbole „+“ und „-“ werden sowohl im Alphabet eines L-Systems als auch bei arithmetischen Operationen in Parameterangaben verwendet, ihre Bedeutung ist abhängig vom Kontext, in dem sie angewendet werden. [PL04, S.46]

Die grafische Turtle-Interpretation einer Zeichenkette, die durch ein L-System zurückgegeben wird, sind somit die Linien, die auf Grundlage der definierten Symbole gezeichnet werden.

Beispiel: Mithilfe von L-Systemen und einer Turtle-Interpretation können Fraktale, in diesem Beispiel sogenannte Koch-Kurven, visualisiert werden. Diese Kurven bestehen aus einem Initiator – einer einfachen, zweidimensionalen Form – und

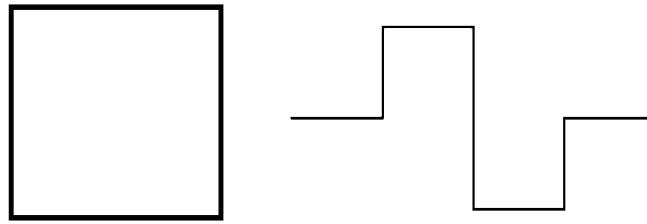


Abb. 2.3: Links: Initiator der Koch-Kurve in Form eines einfachen Quadrats. Rechts: Generator der Koch-Kurve in Form eines offenen Polygonzugs. Eigene Abbildung.

einem Generator, der einem offenen Polygonzug entspricht. In jedem Ableitungsschritt, angefangen bei dem Initiator, wird jede gerade Linie durch den Generator ersetzt. [Man83, S.39]

Dieses Verhalten kann nun auf ein L-System abgebildet werden: Der Initiator entspricht dem Axiom und der Generator einer Produktionsregel des L-Systems. Der in Abbildung 2.3 dargestellte Initiator und Generator entsprechen der Turtle-Interpretation des folgenden L-Systems:

$$\begin{aligned}\omega &: F - F - F - F \\ p_1 &: F \rightarrow F + F - F - FF + F + F - F\end{aligned}\tag{2.6}$$

Für eine bessere Übersicht wurde die Angabe der Parameter weggelassen. Die Turtle interpretiert F als $F(l)$, $-$ als $-(d)$ und $+$ als $+(d)$ mit festgelegter Strichlänge l und Drehwinkel d . Die Entwicklung des L-Systems läuft, als Turtle-Interpretation visualisiert, wie in Abbildung 2.4 dargestellt ab.

2.3.2 Verzweigte L-Systeme

Die bisherigen Definitionen von L-Systemen und die korrespondierende Turtle-Interpretation lässt lediglich die Bildung von Grafiken zu, die einen einzelnen, zusammenhängenden Polygonzug erlauben. Um L-Systeme zu bilden, deren Visualisierungen Baumstrukturen ähneln muss die bisherige Turtle-Interpretation um die Möglichkeit erweitert werden, Verzweigungen zu verarbeiten. [PL04, S.24]

Folgende Operationen werden eingeführt:

- [Der aktuelle Zustand der Turtle in Form ihrer Position und Rotation wird auf einem Kellerspeicher (Stack) abgelegt.
-] Der oberste Zustand der Turtle wird vom Kellerspeicher genommen. Die aktuelle Position und Rotation der Turtle wird auf die im Zustand gespeicherte Position und Rotation gesetzt. Es wird keine Linie zwischen der alten und neuen Position gezeichnet.

¹ „Turtle“ – englisch für „Schildkröte“.

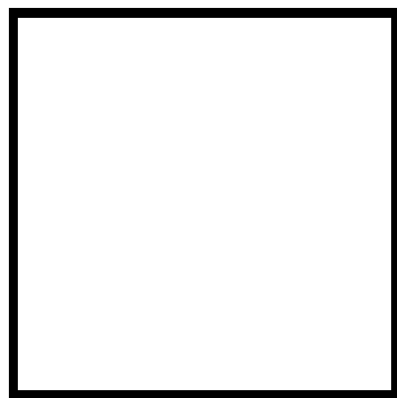
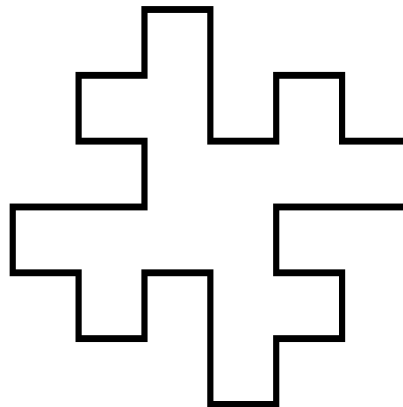
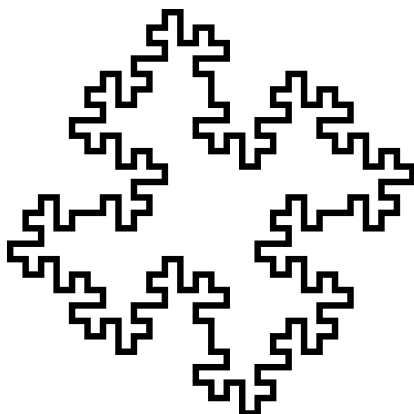
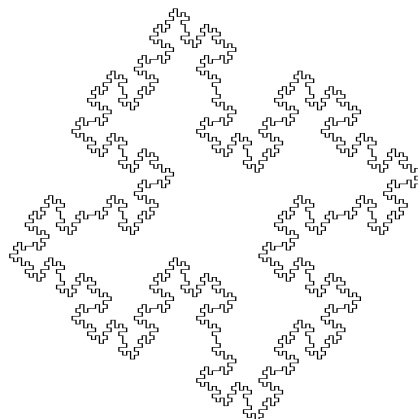
(a) $n = 0, l = 400, d = 90^\circ$ (b) $n = 1, l = 100, d = 90^\circ$ (c) $n = 2, l = 25, d = 90^\circ$ (d) $n = 3, l = 6.25, d = 90^\circ$

Abb. 2.4: Die n -fache Ableitung des Axioms ω anhand der Produktionsregel p_1 aus Gleichung 2.6, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildung.

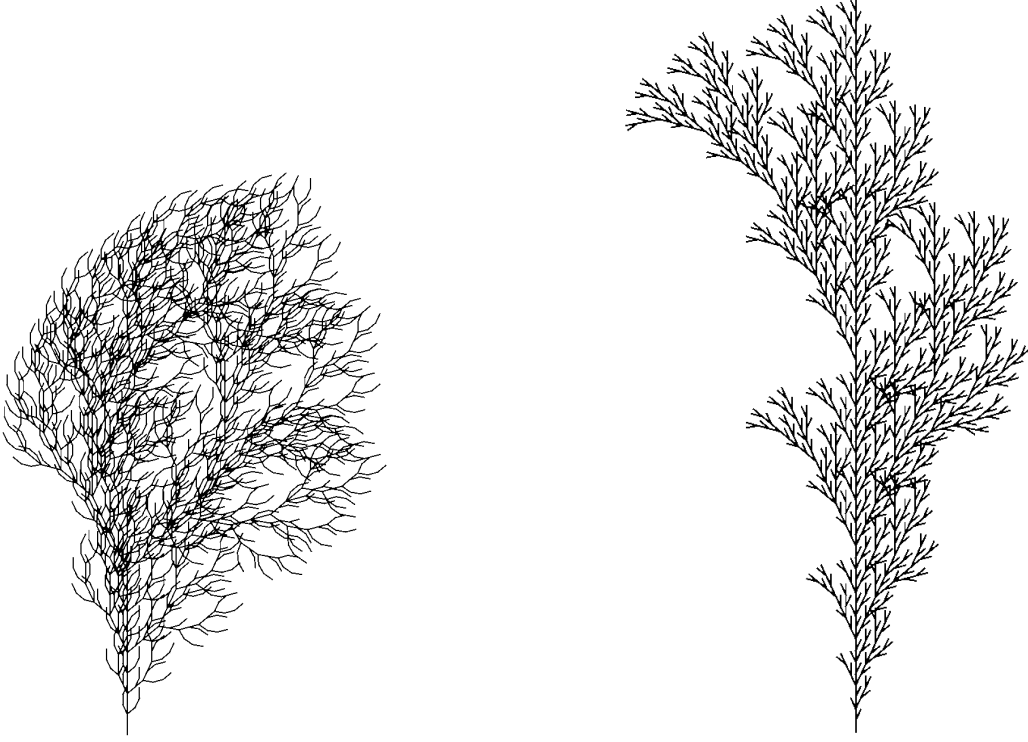
[PL04, S.24]

Diese Erweiterung erlaubt es mehrere Linien zu zeichnen, die von einem einzigen Punkt ausgehen – die Visualisierung von Abzweigungen ist somit möglich. Die Operatoren `[` und `]` markieren den Anfang und das Ende eines Zweiges. [PL04, S.24]

Beispiel: Mithilfe von verzweigten L-System-Beschreibungen lassen sich die in Abbildung 2.5 gezeigten Strukturen bilden. Die Turtle-Interpretation folgt der in 2.3.1 beschriebenen Interpretation mit fester Strichlänge l und festem Drehwinkel d .

2.3.3 Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum

Die bisherige Definition eines Turtle-Zustands mithilfe einer zweidimensionalen Position und einer Blickrichtung genügt nicht, um Visualisierungen von L-

(a) $n = 4, l = 18, d = 25^\circ$

$\omega : F$
 $p_1 : F \rightarrow FF - [-F + F + F] + [+F - F - F]$
 [PL04, S.25]

(b) $n = 5, l = 15, d = 25^\circ$

$\omega : F$
 $p_1 : F \rightarrow F[-F]F[+F][F]$
 [STN16, S.78]

Abb. 2.5: Die n -fache Ableitung der Axiome anhand der Produktionsregeln, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildung.

Systemen in Form von dreidimensionalen Baumstrukturen zu ermöglichen. Sowohl die Zustands-Definition als auch die interpretierten Operationen müssen angepasst und erweitert werden.

Der Zustand der Turtle im dreidimensionalen Raum besitzt eine Position \vec{p} sowie eine 3×3 Rotationsmatrix \mathbf{R} , welche die Orientierung der Turtle im Raum beschreibt. Der Zustand wird als Tupel (\vec{p}, \mathbf{R}) angegeben. \mathbf{R} entspricht zu Anfang einer Identitätsmatrix. Die Einheitsvektoren H , L und U sind orthogonal zueinander und bilden das lokale Koordinatensystem der Turtle.

\vec{H}

Die Blickrichtung (Heading-Vektor) der Turtle. Eine Rotation um diesen Vektor um den Winkel d entspricht der Rotationsmatrix $R_{\vec{H}}$:

$$R_{\vec{H}}(d) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(d) & \sin(d) \\ 0 & -\sin(d) & \cos(d) \end{pmatrix} \quad (2.7)$$

\vec{L}

Der Vektor, der im lokalen Koordinatensystem der Turtle nach links zeigt (Left-Vektor). Eine Rotation um diesen Vektor um den Winkel

d entspricht der Rotationsmatrix $R_{\vec{L}}$:

$$R_{\vec{L}}(d) = \begin{pmatrix} \cos(d) & 0 & \sin(d) \\ 0 & 1 & 0 \\ -\sin(d) & 0 & \cos(d) \end{pmatrix} \quad (2.8)$$

\vec{U}

Der Vektor, der im lokalen Koordinatensystem der Turtle nach oben zeigt (Up-Vektor). Eine Rotation um diesen Vektor um den Winkel d entspricht der Rotationsmatrix $R_{\vec{U}}$:

$$R_{\vec{U}}(d) = \begin{pmatrix} \cos(d) & -\sin(d) & 0 \\ \sin(d) & \cos(d) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.9)$$

[PL04, S.19] [DL05, S.69]

Die erweiterte Turtle-Interpretation verarbeitet folgende Symbole:

$F(l)$ Die Turtle bewegt sich um $l > 0$ in Blickrichtung \vec{H} . Die neue Position ist \vec{p}_{neu} mit:

$$\vec{p}_{neu} = \vec{p} + \mathbf{R} * \vec{H} * l \quad (2.10)$$

Zwischen der alten Position p und der neuen Position p_{neu} wird eine Linie gezeichnet.

$+(d)$ Die Turtle dreht sich nach links um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{U}}(d) \quad (2.11)$$

$-(d)$ Die Turtle dreht sich nach rechts um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{U}}(-d) \quad (2.12)$$

$\&(d)$ Die Turtle neigt sich nach unten um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{L}}(d) \quad (2.13)$$

$^{\wedge}(d)$ Die Turtle neigt sich nach oben um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{L}}(-d) \quad (2.14)$$

$\backslash(d)$ Die Turtle rollt sich nach links um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{H}}(d) \quad (2.15)$$

$/(d)$ Die Turtle rollt sich nach rechts um den Winkel d . Die neue Rotationsmatrix entspricht \mathbf{R}' mit:

$$\mathbf{R}' = \mathbf{R} * R_{\vec{H}}(-d) \quad (2.16)$$

[PL04, S.19] [DL05, S.69]

Mithilfe der Turtle-Interpretation im dreidimensionalen Raum können nun L-Systeme visualisiert werden, die Baumstrukturen ähneln. Ein Beispiel dafür ist das L-System aus Gleichung 2.17, dargestellt in Abbildung 2.6.

$$\begin{aligned} \omega &: /(45) A \\ p_1 &: A \quad \rightarrow F(50) [\&(a)F(50)A] /(d) [\&(a)F(50)A] /(d) [\&(a)F(50)A] \\ p_2 &: F(l) \quad \rightarrow F(l * l_r) \end{aligned} \quad (2.17)$$

[PL04, S.60]

2.4 Anpassungen für Baumstrukturen

Die Darstellung von verzweigten L-Systemen mithilfe von Zeichenketten ermöglicht eine Repräsentation in Textform und erlaubt die einfache Definition von Produktionsregeln. Um jedoch eine effiziente Nachbearbeitung und Visualisierung der Ergebnisse eines L-Systems zu ermöglichen, müssen die Aktionen einer Turtle als graphentheoretischer Baum festgehalten werden. Weiterhin wird der Begriff des Tropismus und dessen Einfluss auf die Modellierung von Baumstrukturen eingeführt.

2.4.1 Graphentheoretische Bäume

Ein Baum, in der Graphentheorie, ist ein kreisfreier Graph $G = \langle V, E \rangle$ für den folgende Begriffe festgelegt werden:

V Die Menge der Knoten. Jeder Knoten entspricht einem Punkt \vec{p}_v im dreidimensionalen Raum. [Sch14, S.358]

E Die Menge der Kanten, welche die Vorgänger-Nachfolger-Beziehungen zwischen den Knoten darstellt. Eine Kante $e \in E$ wird als Tupel $e = (v_1, v_2)$ mit $v_1, v_2 \in V$ dargestellt. v_1 wird als Vorgänger von v_2 und v_2 als Nachfolger von v_1 bezeichnet. Jeder Knoten besitzt maximal einen Vorgänger und eine endliche Menge von Nachfolgern. [Sch14, S.358] [Lux14, S.29]

Wurzel Eine Wurzel ist ein Knoten $v \in V$ ohne einen Vorgänger. Der Baum besitzt genau eine Wurzel. [Sch14, S.358]

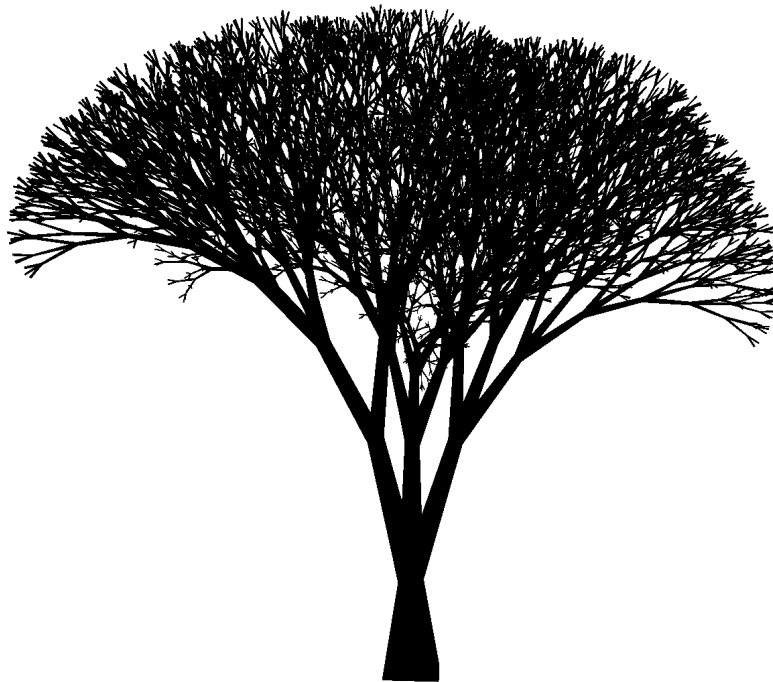
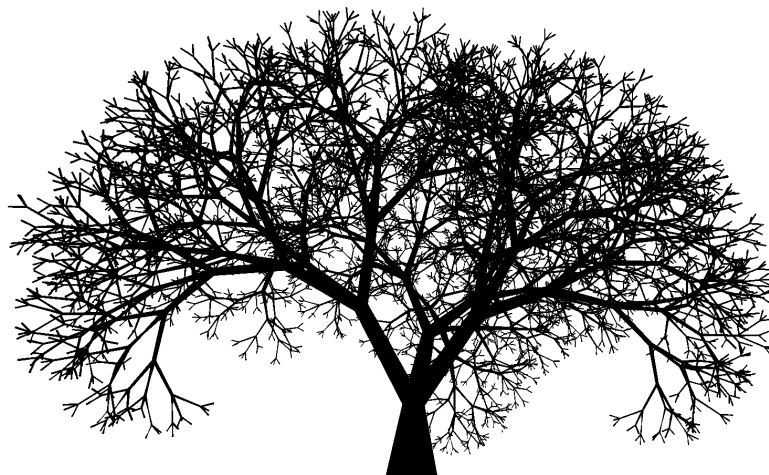
(a) $n = 8$, $d = 137.5^\circ$, $a = 18.95^\circ$, $l_r = 1.3$ (b) $n = 8$, $d = 137.5^\circ$, $a = 36.0^\circ$, $l_r = 1.3$

Abb. 2.6: Die n -fache Ableitung des Axioms anhand der Produktionsregeln aus Gleichung 2.17, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildungen.

Grad Der Grad eines Knotens ist definiert als die Anzahl seiner Nachfolger. [Lux14, S.29]

Tiefe Die Tiefe eines Knotens ist definiert als die Länge der Folge von Vorgängern, die durchlaufen werden muss, bis die Wurzel erreicht wurde. Die Wurzel besitzt die Tiefe 0. [Lux14, S.30]

Ein Baum ist grundsätzlich ein rein topologisches Objekt, mithilfe der Zuordnung von Punkten zu Knoten kann jedoch eine geometrische Vorstellung des Aufbaus bewirkt werden. [PL04, S.23]

Um den Aufbau eines Baumes $G = \langle V, E \rangle$ mithilfe der Turtle-Interpretation zu ermöglichen, muss der Turtle-Zustand um einen Knotenpunkt $v \in V$ ergänzt werden. Der Zustand wird als Tupel \vec{p}, v, \mathbf{R} angegeben.

Die Turtle-Interpretation beginnt mit der Erstellung der Wurzel am Ursprung des lokalen Bezugssystems. Die Verarbeitung folgender Symbole muss erweitert werden:

- F(l)** Die neue Position \vec{p}_{neu} der Turtle wird entsprechend Gleichung 2.10 berechnet. An diesem Punkt wird ein Knoten v_{neu} und eine Kante (v, v_{neu}) erstellt sowie v als Vorgänger von v_{neu} und v_{neu} als Nachfolger von v eingetragen. v entspricht dem aktuellen Zustandsknoten der Turtle. Der neue Turtle-Zustand ist $(\vec{p}_{neu}, v_{neu}, \mathbf{R})$.
- [Zusätzlich zu der Position und Rotation der Turtle wird auch der aktuelle Knoten auf einem Kellerspeicher abgelegt.
-] Der oberste Zustand der Turtle wird vom Kellerspeicher genommen. Zusätzlich zur Position und Rotation wird auch der Zustandsknoten der Turtle auf den gespeicherten Knoten gesetzt.

Die Kanten zwischen Knoten wurden durch die Turtle-Interpretation bisher als Linien visualisiert und können als Astsegmente von echten Baumstrukturen verstanden werden. [PL04, S.23] Die Repräsentation von Turtle-Aktionen als graphentheoretischer Baum ermöglicht jedoch die in Abschnitt 4.4 beschriebene Generierung und Nachbearbeitung von Modelldaten.

2.4.2 Tropismus

Tropismus ist die Tendenz einer Pflanze in eine bestimmte Richtung zu wachsen, beispielsweise aufgrund einer Lichtquelle oder der Beugung durch Gravitation. [RLP07, Abschn. 3] Der Einfluss von Tropismus wird als ein dreidimensionaler Vektor \vec{T} angegeben. Die Bewegung $F(l)$ der Turtle wird wie folgt erweitert:

- F(l)** Die Turtle bewegt sich um $l > 0$ in Blickrichtung \vec{H} . Die neue Position ist \vec{p}_{neu} mit:

$$\vec{p}_{neu} = \vec{p} + l * \frac{\vec{H}_{rot} + e * \vec{T}}{\|\vec{H}_{rot} + e * \vec{T}\|} \quad (2.18)$$

$$\text{und } \vec{H}_{rot} = \mathbf{R} * \vec{H} \quad (2.19)$$

wobei e der Anfälligkeit des Baums für die Beugung durch Tropismus entspricht. [PL04, S.58] [RLP07, Abschn. 3]

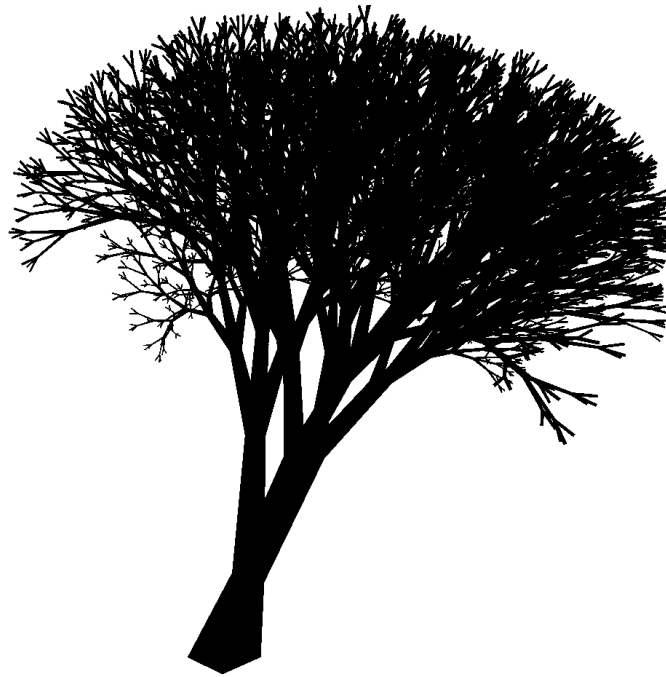


Abb. 2.7: .

$$n = 8, d = 137.5^\circ, a = 36.0^\circ, l_r = 1.3, \vec{T} = \begin{pmatrix} 0 \\ 1 \\ -0.5 \end{pmatrix}, e = 0.27$$

Die n-fache Ableitung des Axioms anhand der Produktionsregeln aus Gleichung 2.17, visualisiert mithilfe der erweiterten Turtle-Interpretation. Zeigt den Einfluss von Tropismus auf den in Abbildung 2.6b dargestellten Baum. Eigene Abbildung.

Space Colonization Algorithmus

Es werden die benötigten Eingaben und der Ablauf des Space Colonization Algorithmus festgelegt. Die Prozedur zur Generierung von Baumstrukturen wird vorgestellt und die implementierten Erweiterungen des ursprünglichen Algorithmus werden erläutert.

3.1 Ursprung

Der Space Colonization Algorithmus wurde ursprünglich zur Modellierung und Visualisierung von Blattvenen vorgeschlagen und basiert auf der Wirkung des Pflanzenhormons Auxin. Dieser Hormonstoff entsteht im Blatt und wird von bereits existierenden Blattvenen angezogen, der resultierende Hormonstrom führt zur Bildung von neuen Venen im Blatt. Die Simulation dieses Vorgangs führt zu realitätsnahen Venenmustern. [RFL⁺05, Abschn. 2.5]

Mithilfe einer Erweiterung in den dreidimensionalen Raum und Nachbearbeitung der Resultate kann eine Vielfalt von Baum- und Strauchstrukturen generiert werden. [RLP07, Abschn. 1]

3.2 Aufbau

Der Algorithmus verarbeitet eine Menge von Einflusspunkten S und baut darauf basierend einen Baum $G = \langle V, E \rangle$ auf.

Der Algorithmus benötigt die folgenden Eingaben:

- d_i Der Einflussradius. Einflusspunkte prägen den Aufbau des Baums nur, wenn sich Knotenpunkte innerhalb dieses Radius befinden. [RLP07, Abschn. 2]
- d_k Der Minimalradius. Befindet sich ein Knotenpunkt innerhalb des Minimalradius um einen Einflusspunkt, wird dieser aus der Menge der Einflusspunkte S entfernt. [RLP07, Abschn. 2]

- D** Die Schrittweite. Jeder neu generierte Knotenpunkt wird in diesem Abstand zu seinem Vorgänger positioniert. [RLP07, Abschn. 2]
- \vec{T}** Der Tropismusvektor.

3.3 Ablauf

Zu Beginn des Algorithmus werden N Einflusspunkte in einem vorgegebenen Bereich generiert. Dieser Einflussbereich signalisiert die Verfügbarkeit von Raum, in dem die Baumstruktur entstehen kann. [RLP07, Abschn. 2]

Daraufhin wird der Baum iterativ aufgebaut und durchläuft in jeder Iteration die folgenden Schritte:

1. Für jeden Einflusspunkt in S wird der am nächsten liegende Knotenpunkt $v \in V$ bestimmt. Befindet sich v innerhalb des Einflussradius d_i um den Einflusspunkt herum, wird dieser einer zugeordneten Menge $S(v)$ hinzugefügt. $S(v)$ beinhaltet somit alle Einflusspunkte, die einen Einfluss auf den Knotenpunkt ausüben. [RLP07, Abschn. 2]
2. Befinden sich Elemente in $S(v)$, wird ein neuer Knotenpunkt v' den Nachfolgern von v hinzugefügt und v als Vorgänger von v' eingetragen. Alle Punkte in $S(v)$ beeinflussen v' in gleichem Maße, die neue Position $\vec{p}_{v'}$ des Knotenpunkts berechnet sich somit wie folgt:

$$\vec{p}_{v'} = \vec{p}_v + D * \tilde{n} \quad (3.1)$$

$$\text{mit } \tilde{n} = \frac{\vec{n} + \vec{T}}{\|\vec{n} + \vec{T}\|} \quad (3.2)$$

$$\text{und } \vec{n} = \sum_{s \in S(v)} \frac{\vec{p}_s - \vec{p}_v}{\|\vec{p}_s - \vec{p}_v\|} \quad (3.3)$$

[RLP07, Abschn. 2]

3. Für jeden Einflusspunkt wird überprüft, ob sich ein Knotenpunkt innerhalb des Minimalradius d_k befindet. Existiert ein solcher Knotenpunkt, wird der Einflusspunkt aus der Menge der Einflusspunkte S entfernt. [RLP07, Abschn. 2]

Diese Schritte werden solange ausgeführt, bis alle Einflusspunkte entfernt wurden, sich kein Knotenpunkt im Einflussradius eines Einflusspunktes befindet oder bis eine vorgegebene Maximalanzahl von Iterationen durchgeführt wurde. Abbildung 3.1 zeigt eine beispielhafte Anwendung des Algorithmus.

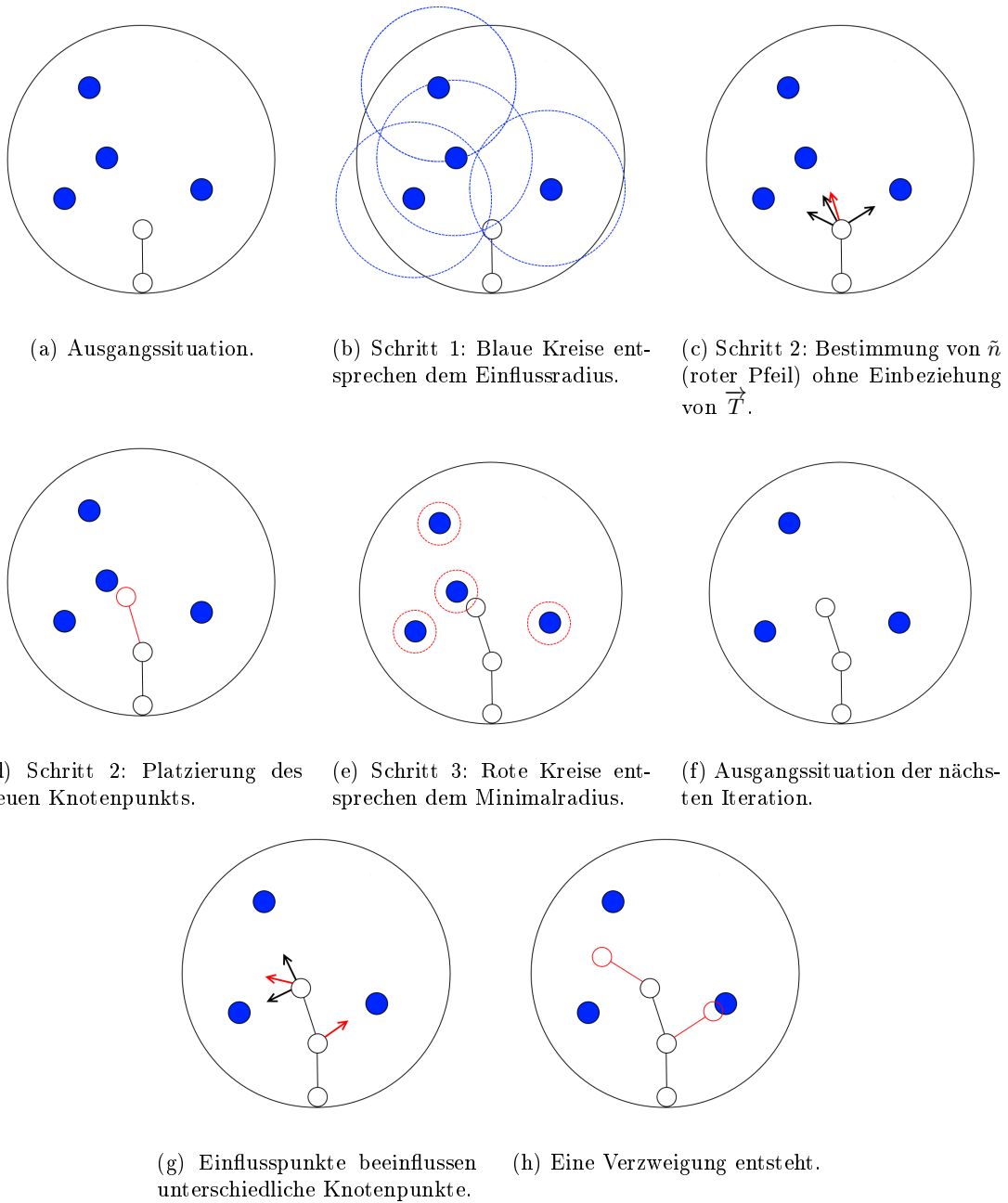


Abb. 3.1: Beispielhafte Anwendung des Space Colonization Algorithmus. Blaue Punkte entsprechen Einflusspunkten, weiße Punkte entsprechen Knotenpunkten. Der unterste Knotenpunkt stellt die Wurzel dar. Eigene Abbildungen auf Grundlage von [RLP07, Abb. 2].

3.4 Aufbau der Baumstrukturen

Der Space Colonization Algorithmus liefert einen Baum, der Knotenpunkte enthält, welche Positionen im dreidimensionalen Raum darstellen. Um diese Knotenpunkte in Form von baumähnlichen Strukturen zu visualisieren, wird die Prozedur erweitert. Der Aufbau der Baumstruktur läuft wie folgt ab:

1. Der Einflussbereich wird mit der vorgegeben Anzahl von Einflusspunkten gefüllt. [RLP07, Abschn. 2]
2. Der Baum wird wie in Abschnitt 3.3 beschrieben iterativ generiert. [RLP07, Abschn. 2]
3. Die Nachfolger jedes Knotenpunkts werden einander angenähert, um eine Verringerung der Abzweigungswinkel zwischen den verbindenden Kanten zu erreichen. Dies führt zu einer insgesamt realistischeren Baumstruktur. [RLP07, Abschn. 2]
4. Die Kanten, welche die Knotenpunkte verbinden, werden mithilfe von Zylindern visualisiert, um die Aststruktur eines biologischen Baumes zu simulieren. [RLP07, Abschn. 2]

Die Berechnung der Zylinderbreiten erfolgt anhand von Murrays Regel, die besagt, dass der Radius r eines Astes auf Grundlage der Radien $r_{n_1} \dots r_{n_m}$ der nachfolgenden, von ihm abzweigenden Äste wie folgt berechnet werden kann:

$$r^g = r_{n_1}^g + r_{n_2}^g + \dots + r_{n_m}^g \quad (3.4)$$

Diese Berechnung kann rekursiv auf dem Baum ausgeführt werden, indem für jeden Knotenpunkt der Radius aus den Radien seiner Nachfolger berechnet wird. Besitzt ein Knotenpunkt keine Nachfolger, wird ein festgelegter Radius r_0 zurückgegeben. [RFL⁺05, Abschn. 3.5] Der Wert g kann frei gewählt werden.

Die von Runions u.a. [RLP07] vorgeschlagene Kurven-Unterteilung [RLP07, Abschn. 2] wurde in dieser Arbeit nicht behandelt, da durch Angabe der Schrittweite D eine ausreichende visuelle Qualität erzielt wurde.

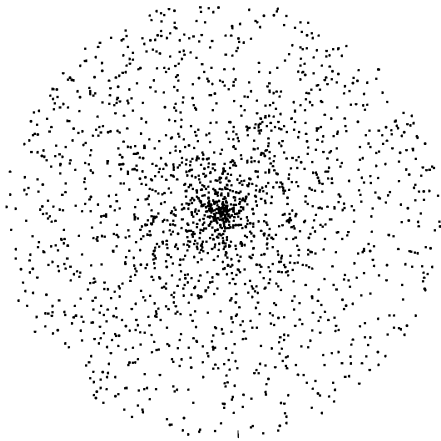
Abbildung 3.2 zeigt die Modellierung einer zweidimensionalen Baumstruktur.

3.5 Erweiterungen

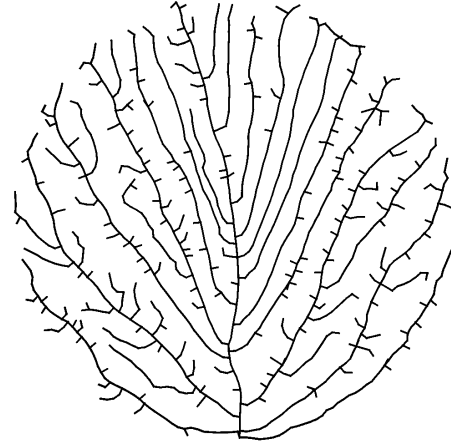
Die folgenden Erweiterungen des ursprünglichen Algorithmus wurden implementiert.

Zweigtiefe

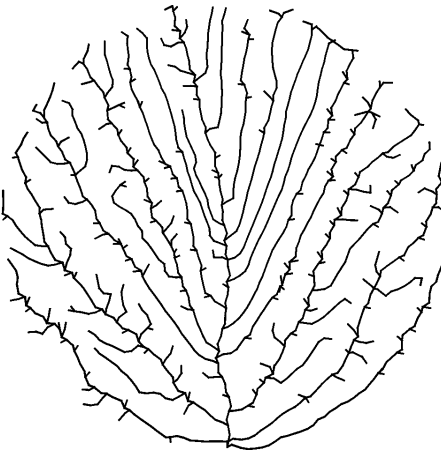
Die Zweigtiefe $Z(v)$ eines Knotens $v \in V$ wird für Erweiterungen des Space Colonization Algorithmus und die Generierung von Modelldaten verwendet. Sie entspricht nicht der Tiefe des Knotens und berechnet sich wie folgt:



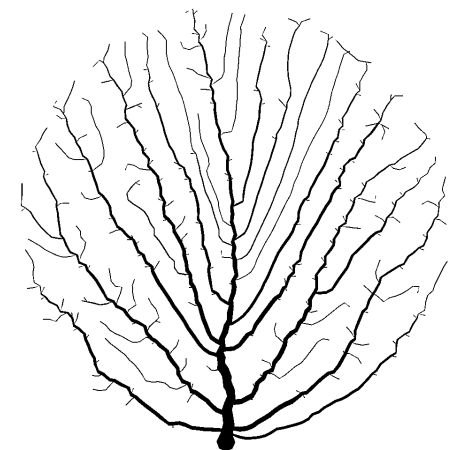
(a) $N = 2000$ Einflusspunkte, zufällig in einem Ring mit dem Radius $r = 500$ verteilt.



(b) Ergebnis des Space Colonization Algorithmus mit einem Einflussradius $d_i = 100$, Minimalradius $d_k = 20$, Schrittweite $D = 15$ und $\vec{T} = \vec{0}$.



(c) Verringerung der Abzweigungswinkel.



(d) Modellierung des Ergebnis mithilfe von Zylindern mit $r_0 = 1$ und $g = 2$.

Abb. 3.2: Modellierung einer zweidimensionalen Baumstruktur entsprechend der in Abschnitt 3.4 beschriebenen Schritte. Eigene Abbildungen.

$$Z(v) = \begin{cases} 0 & \text{falls } v \text{ die Wurzel ist} \\ Z(v') & \text{falls } v' \text{ genau einen Nachfolger besitzt} \\ 1 + Z(v') & \text{sonst} \end{cases} \quad (3.5)$$

wobei v' den Vorgänger des Knotens v darstellt. Die Zweigtiefe wird direkt nach der Erstellung des Knotens berechnet und daraufhin nicht verändert. Ein Beispiel für die Bestimmung der Zweigtiefe wird in Abbildung 3.3 gezeigt.

Der von Prusinkiewicz u.a. [PL04] beschriebene Begriff von axialen Bäumen [PL04, S.21] und die zugehörige Bestimmung der Achsentiefe unterscheidet sich von der Berechnung der Zweigtiefe. Knoten können dieselbe Zweigtiefe besitzen, selbst wenn ihre Positionen keine gerade Linie bilden.

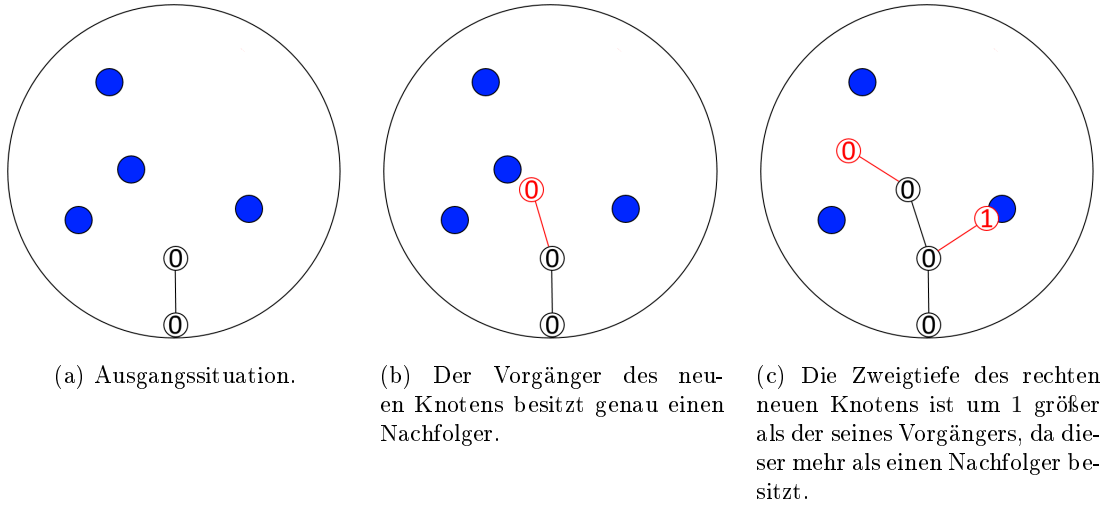


Abb. 3.3: Darstellung der Zweigtiefen von Knotenpunkten aus Abbildung 3.1.

Zusätzliche Bedingungen

Um mehr Kontrolle über die vom Space Colonization Algorithmus generierte Baumstruktur zu bekommen, wurden zusätzliche Bedingungen an den Ablauf des Algorithmus hinzugefügt:

max_{grad} Der maximale Grad der Knotenpunkte im Baum. Bevor ein neuer Knotenpunkt v_n als Nachfolger eines Knotenpunkts v erstellt wird, wird die Anzahl der Nachfolger von v überprüft. Entspricht diese max_{grad} , wird der Knotenpunkt v_n nicht erstellt.

max_Z Die maximale Zweigtiefe eines Knotens. Bevor ein neuer Knotenpunkt v' als Nachfolger eines Knotenpunkts v erstellt wird, wird seine Zweigtiefe $Z(v')$ berechnet. Übersteigt diese max_Z , wird der Knotenpunkt v' nicht erstellt.

max_{NG} Die maximale Anzahl von Iterationen, in welchen einem Knotenpunkt kein neuer Nachfolger hinzugefügt wurde.¹ Hat ein Knoten diesen Wert erreicht, wird er in Schritt 1 des Algorithmus 3.3 nicht weiter darauf untersucht, ob er sich innerhalb des Einflussradius eines Einflusspunktes befindet. Der Wert von max_{NG} eines Knotenpunkts wird auf 0 zurückgesetzt, falls diesem ein neuer Nachfolger hinzugefügt wurde. Durch die vorsichtige Wahl von max_{NG} können Positionsvergleiche vermieden werden ohne den resultierenden Baum zu verändern.

Gewichtetes Wachstum

Gewichtetes Wachstum ist eine Möglichkeit die Schrittweite D in Abhängigkeit von der Zweigtiefe zu erhöhen. Befindet sich ein Knotenpunkt auf der Zweigtiefe

¹ NG steht für „No Grow“ oder „Did not grow“, englisch für „kein Wachstum“.

0, werden neu hinzugefügte Nachfolger im Abstand von $2 * D$ positioniert, befindet er sich auf der maximalen Zweigtiefe, werden neu hinzugefügte Nachfolger im Abstand von D positioniert. Zwischen diesen beiden Zweigtiefen wird die Schrittweite linear interpoliert.

Kurvenreduktion

Kurvenreduktion in Abhängigkeit des Abzweigungswinkels ermöglicht es die Baumstruktur mithilfe einer verringerten Datenmenge darzustellen. Besitzt ein Knoten v genau einen Nachfolger v_n und einen Vorgänger v' , werden die zwei Richtungsvektoren zwischen v' und v sowie v und v_n berechnet:

$$\vec{R}_1 = \frac{p_v - p_{v'}}{\|p_v - p_{v'}\|} \text{ und } \vec{R}_2 = \frac{p_{v_n} - p_v}{\|p_{v_n} - p_v\|} \quad (3.6)$$

Übersteigt das Skalarprodukt $\langle \vec{R}_1, \vec{R}_2 \rangle$ einen festgelegten Maximalwert, wird v aus dem Baum entfernt. v_n wird zum Nachfolger von v' und v' zum Vorgänger von v_n .

Implementierung

Die Implementierung des Projekts lässt sich in vier Bereiche unterteilen: Die Baumrepräsentation enthält Daten, die von den L-System und Space Colonization Implementierungen generiert werden. Diese Daten werden an das Modellgenerierungssystem übergeben, welches die Modelldaten für eine grafische Darstellung in der Unreal Engine 4 produziert.

4.1 Baumrepräsentation

Sowohl L-Systeme als auch der Space Colonization Algorithmus generieren einen graphentheoretischen Baum, auf Grundlage dessen die Modellgenerierung durchgeführt wird. Die implementierte Baumrepräsentation kann daher von beiden Systemen verwendet werden und ermöglicht es, diese mit demselben Modellgenerierungssystem zu visualisieren.

Die Baumstruktur wird durch eine Datenklasse repräsentiert, jedes Objekt dieser Klasse beschreibt einen Knoten sowie die Kante, welche vom Vorgänger zu dem Knoten führt. Die Datenklasse bietet Zugriff auf die folgenden Informationen:

Vorgänger und Nachfolger: Mithilfe eines Verweises auf den Vorgänger und eine Liste der Nachfolger eines Knotens kann der Baum-Graph vollständig repräsentiert werden. Weiterhin ermöglicht dies die Implementierung einer Reihe von rekursiven Funktionen zur Anpassung von Modelldaten.

Modell-Daten: Kanten werden, wie in Abschnitt 4.4.3 beschrieben, mithilfe von Zylindern visualisiert. Um die Generierung von Modelldaten zu vereinfachen, bietet die Datenklasse Zugriff auf Start- und Endposition, Start- und Endradius, Start- und Endnormale sowie einen Rotationswinkel.

Weiterhin wird die Zweigtiefe des repräsentierten Knoten gespeichert.

Wachstums-Daten: Die Wachstums-Daten bestehen aus einer Wachstumsrichtung, einem Einfluss-Zähler und dem „Kein Wachstum“-Zähler (*NG*-Counter), welche für den Ablauf des Space Colonization Algorithmus benötigt werden.

Ein Objekt der Datenklasse kann als Astsegment eines biologischen Baumes angesehen werden und wird durch das Modellgenerierungssystem als solches visualisiert.

4.2 L-Systeme

Die Implementierung einer L-System-Baumstruktur wird durch einen Unreal-Akteur verwirklicht, der im Level platziert werden kann. Nach Start des Levels wird das angegebene Axiom anhand der Produktionsregeln abgeleitet und die sich ergebende Zeichenkette von der Turtle-Implementierung interpretiert.

4.2.1 Parameter

Dem L-System-Akteur werden die folgenden Parameter über die Editor-UI übergeben:

Anzahl der Ableitungen: Die Anzahl der Ableitungen in \mathbb{N}^+ , die auf dem Axiom durchgeführt werden sollen.

Axiom: Das Axiom in Form einer Zeichenkette.

Konstanten: Eine Konstante besteht aus der Angabe eines Identifikationssymbols und eines Wertes in \mathbb{R} . Konstanten können im Axiom und in den Nachfolgern der Produktionen verwendet werden.

Produktionen: Jede Produktion besteht aus Angabe eines Vorgängers, eine Liste von Parametern und einem Nachfolger. Der Vorgänger und jeder Parameter entspricht einem einzelnen Symbol, der Nachfolger wird als Zeichenkette eingetragen. Die Parametersymbole können nur innerhalb des Nachfolgers verwendet werden. Der Vorgänger und die Liste der Parameter bilden das parametrische Wort, welches bei einer Ableitung durch den Nachfolger ersetzt wird.

Tropismus: Der Einfluss von Tropismus in Form eines dreidimensionalen Vektors und eines Biegsamkeitswertes.

Für das Axiom und die Produktionen gelten die in Kapitel 2 festgelegten Regeln für die Definition von L-Systemen. Weiterhin gelten die üblichen Regeln für die Angabe von arithmetischen Operationen. Die Verwendung von Klammern ist jedoch auf die Angabe von Parametern eines parametrischen Wortes beschränkt, ihre Verwendung zur Beeinflussung der Auswertungsreihenfolge eines arithmetischen Ausdrucks wird nicht unterstützt.

Die Parameter für eine Anpassung der Visualisierung der Turtle-Interpretationen werden in Abschnitt 4.4.1 erläutert.

Ein Beispiel für die korrekte Eingabe eines L-Systems über die Editor-UI wird in Abbildung 4.1 gezeigt.

The screenshot shows a configuration window for an L-System. The left sidebar contains a tree view with the following structure:

- LSystem
 - LSystem Data
 - Number Of Derivations: 5
 - Axiom: F()
 - Constants
 - 0
 - ID: l
 - Constant Value: 15.0
 - 1
 - ID: d
 - Constant Value: 25.0
 - Productions
 - 0
 - ID: F
 - Parameter List
 - 0
 - Production Result

The right pane displays the configuration details for the selected '0' production rule:

- 2 Array elements (with +, trash, and copy icons)
- 2 members (dropdown menu)
- ID: l
- Constant Value: 15.0
- 2 members (dropdown menu)
- ID: d
- Constant Value: 25.0
- 1 Array elements (with +, trash, and copy icons)
- 3 members (dropdown menu)
- ID: F
- 1 Array elements (with +, trash, and copy icons)
- ID: l
- Production Result: F()[-(d)F()F()[(d)F()]]F()

Abb. 4.1: Ein Beispiel für die Angabe des L-Systems aus Gleichung 2.5b mit der resultierenden Baumstruktur aus Abbildung 2.5b.

4.2.2 Ableitung

Zu Anfang der Erstellung des L-System-Akteurs werden alle Konstantensymbole im Axiom und den Produktionen durch die Konstantenwerte ersetzt.

Die Implementierung arbeitet durchgehend auf derselben Zeichenkette, angefangen mit dem Axiom. In jeder Ableitung werden die in den Produktionsregeln definierten parametrischen Wörter durch die angegebenen Nachfolger ersetzt.

Nachdem die vorgegebene Anzahl von Ableitungen durchgeführt wurde, wird die resultierende Zeichenkette an die Turtle-Implementierung weiter gegeben.

4.2.3 Turtle Interpretation

Die Turtle-Implementierung bekommt die aus den Ableitungen resultierende Zeichenkette übergeben. Diese wird daraufhin sequentiell abgearbeitet und entsprechend den in Abschnitt 2.4 vorgestellten Konzepten interpretiert. Der resultierende Baum wird für die Konstruktion des Modells an das Modellgenerierungssystem weitergegeben.

4.3 Space-Colonization Algorithmus

Die Implementierung einer Space-Colonization-Baumstruktur stellt sich aus der Platzierung von mindestens einem Akteur für die Repräsentation des Einflussbereichs und einem Akteur für die Umsetzung des Algorithmus zusammen.

4.3.1 Parameter

Dem Space-Colonization-Akteur werden Parameter des ursprünglichen Algorithmus sowie die Eingaben für in Abschnitt 3.5 besprochene Erweiterungen über die Editor-UI übergeben. Zu den ursprünglichen Parametern gehören der Minimalradius, Einflussradius, Schrittweite, ein Tropismusvektor sowie die Anzahl der durchzuführenden Iterationen. Zu den erweiterten Parametern gehören der maximale Grad, die maximale Zweigtiefe, die maximale Anzahl von „Kein Wachstum“-Iterationen, der Kurvenreduktionswert und eine Abfrage, ob gewichtetes Wachstum durchgeführt werden soll.

Weiterhin müssen die zugeordneten Einflussbereich-Akteure angegeben werden – damit der Algorithmus durchgeführt werden kann, muss mindestens einer dieser Akteure mit mindestens einem Einflusspunkt eingetragen werden.

4.3.2 Einflussbereich

Pro Einflussbereich:

- Anzahl von Einfluss punkten

- Random Seed

- Zwei Arten von Einflussbereichen: Kugel- und Zylinder-förmig.

- Angabe mehrerer Einflussbereiche möglich

4.3.3 Ablauf des Algorithmus

test

4.4 Modellgenerierung

test

4.4.1 Parameter

Wichtig: Sowohl SCA als auch LS übergeben Werte wie Branch Radius Zero etc.

4.4.2 Procedural Mesh Component

test

4.4.3 Generierung der Zylinder-Meshes

Die Startposition entspricht der Endposition des Vorgängers und markiert den Anfang der Kante, die zu dem Knoten führt, der durch das Datenobjekt beschrieben wird. Die Endposition beschreibt die Position des Knotens und markiert das Ende der Kante, die zu dem Knoten führt.

Der Startradius bestimmt den Radius des Start-Zylinderrings, Startposition und Startnormale beschreiben die Ebene, auf welcher der Zylinderring generiert wird. Der Endradius entspricht dem Radius des End-Zylinderrings, Endposition und Startnormale beschreiben die Ebene, auf welcher der Ring erstellt wird.

4.4.4 Operationen auf der Baumstruktur

Was kann mit BranchUtility gemacht werden?

Ergebnisse

test

5.1 L-System-Akteur

test

5.1.1 Monopodial

test

5.1.2 Simpodial

test

5.1.3 Ternäre Verzweigungen

test

5.1.4 Tropismus

test

5.1.5 Performanz

test

5.2 Space-Colonization-Akteur

test

5.2.1 Ursprüngliche Parameter

test

5.2.2 Erweiterte Parameter

test

5.2.3 Performanz

test

5.3 Probleme

Mit SCA: Gleicher Abstand von zwei Einflusspunkten führt zu beinahe unendlicher Hin- und Her-Generierung von Branches.

SCA: Random-Verteilung der Punkte.

Zusammenfassung und Ausblick

Zusammenfassung

6.1 Erweiterungen

test

6.1.1 Space Colonization Algorithmus

test

Positionsabfragen

Verbesserte Positionsvergleiche bei Einflusspunkt- zu Knotenpunkt-Abfragen.

Einflussbereiche

Verbesserte Möglichkeit, Einflussbereiche anzugeben. Oberfläche, Random-Verteilung

6.1.2 Aststruktur

Generalized cylinders.

6.1.3 Texturen

test

6.1.4 Blätter

test

6.1.5 Generierung zur Laufzeit

test

6.1.6 Verteilung

test

6.1.7 Instanziierung

Eine beschränkte Anzahl Pflanzen generieren und diese so verteilen, dass es nicht bemerkt wird, dass es immer die selben Pflanzen sind

6.1.8 title

test

Literaturverzeichnis

- DL05. DEUSSEN, OLIVER und BERND LINTERMANN: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer-Verlag Berlin Heidelberg 2005, 2005.
- GSJ04. GOLDMAN, RON, SCOTT SCHAEFER und TAO JU: *Turtle Geometry in Computer Graphics and Computer Aided Design*, 2004. <http://www.cs.wustl.edu/~taoju/research/TurtlesforCADRevised.pdf>.
- Lux14. LUX, PROF. DR. ANDREAS: *Algorithmen und Datenstrukturen - Vorlesungsskript Kapitel 4*, 2014.
- Man83. MANDELBROT, BENOIT B.: *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1983.
- PL04. PRUSINKIEWICZ, PRZEMYSŁAW und ARISTID LINDENMAYER: *The Algorithmic Beauty of Plants*. Przemysław Prusinkiewicz, eBook Auflage, 2004. <http://algorithmicbotany.org/papers/abop/abop.pdf>.
- RFL⁺05. RUNIONS, ADAM, MARTIN FUHRER, BRENDAN LANE, PAVOL FEDERL, ANNE-GAËLLE ROLLAND-LAGAN und PRZEMYSŁAW PRUSINKIEWICZ: *Modeling and visualization of leaf venation patterns*, 2005. <http://algorithmicbotany.org/papers/venation.sig2005.pdf>.
- RLP07. RUNIONS, ADAM, BRENDAN LANE und PRZEMYSŁAW PRUSINKIEWICZ: *Modeling Trees with a Space Colonization Algorithm*, 2007. <http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf>.
- Sch14. SCHMITZ, PROF. DR. HEINZ: *Theoretische Informatik - Vorlesungsskript*, 2014.
- STN16. SHAKER, NOOR, JULIAN TOGELIUS und MARK J. NELSON: *Procedural Content Generation in Games*. Springer International Publishing Switzerland 2016, 2016.

A

Glossar

Mesh Englisch für Polygonnetz.

B

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist ...

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser: ...

Datum

Unterschrift der Kandidatin / des Kandidaten